ALTERNATIVE ANALYSIS-SYNTHESIS APPROACHES FOR TIMESCALE, FREQUENCY AND OTHER TRANSFORMATIONS OF MUSICAL SIGNALS

Victor Lazzarini

Department of Music
National University of Ireland, Maynooth
Victor.Lazzarini@nuim.ie

Joe Timoney, Tom Lysaght

Department of Computer Science
National University of Ireland, Maynooth
Joe.Timoney@cs.nuim.ie
Tom.Lysaght@cs.nuim.ie

ABSTRACT

This article presents new spectral analysis-synthesis approaches to musical signal transformation. The analysis methods presented here involve the use of a superior quality technique of frequency estimation, the Instantaneous Frequency Distribution (IFD), and partial tracking. We discuss the theory behind the IFD, comparing it to other existing methods. The partial tracking analysis employed in this process is explained fully. This is followed by a look into the three resynthesis methods proposed by this work, based on different approaches to additive synthesis. A number of transformations of musical signals are proposed to take advantage of the analysis-synthesis techniques. Performance details and specific aspects of this implementation are discussed. This is complemented by a look at some of the results of these methods in the time-stretching of audio signals, where they will be shown to perform better than many of the currently available techniques.

1. INTRODUCTION

Analysis-resynthesis techniques have been an old favourite for transformation processes that involve frequency-independent timescale modifications, time-independent frequency modifications and a number of other interesting techniques[1]. A widespread method of spectral analysis-resynthesis is found in the phase vocoder algorithm[2]. The process uses the discrete Short-Time Fourier Transform (STFT), followed by polar conversion and instantaneous frequency estimation by a difference method. Timescale modification is achieved by resynthesis at a different time rate. This will use either an inverse process of phase integration, followed by rectangular conversion and the inverse transform, or by additive synthesis. If the latter is used, direct frequency transformations such as pitch shifting (scaling), stretching or warping can be implemented. If, instead we use the ISTFT method, then frequency modifications can be achieved by scaling, stretching or warping of the instantaneous frequencies, followed by bin reallocation. Other methods that manipulate the amplitude-frequency data can also be implemented, such as cross-synthesis, interpolation (morphing) and filtering.

Depending on the source signal and on the transformation used amount, the phase vocoder technique will produce an acceptable result. However, due to problems involving phase recuperation[3], artifacts will be produced, which in some cases might result in a poor quality output. A number of approaches have been proposed to minimise this effect, mainly involving phase-locking[4].

This article describes an alternative method of analysis-resynthesis spectral processing. We present here a more accurate method of instantaneous frequency estimation, the IFD and three alternative resynthesis methods. Two of them will accept input from a partial tracking analysis process and the other will resynthesise bin frame data, identical to the phase vocoder amplitude-instantaneous frequency format. The advantages of these techniques lie in the improved quality of the resynthesis, in terms of artifact minimisation and transient response.

2. FREQUENCY ESTIMATION

The method of frequency estimation used in this work is given by the instantaneous frequency distribution (IFD) algorithm. Based on the reassignment theory, this was proposed, originally, by Friedman in [5] and then later, in a slightly different form, by Toshihiko Abe[6]. The principle behind it is that the instantaneous frequency is the time derivative of the phase. With the Discrete Fourier Transform (DFT) in polar form (with $\omega = 2\pi k/N$), we can define the IFD as:

$$DFT(x(n),k,t) = R(\omega,t) \times e^{j\theta(\omega,t)} \qquad (1)$$

$$IFD(x(n),k,t) = \frac{\partial}{\partial t}\theta(\omega,t) \qquad (2)$$

In the phase vocoder, this is estimated using a frame-by-frame difference method. The IFD calculates the time derivative of the phase directly, from data corresponding to a single time-point. As with the phase vocoder, the frequency estimation will use the discrete STFT, taken as a series of DFTs of a rotated and windowed input signal:

$$STFT(x(n),k,t) = e^{-jwt} DFT(x_t(m),k) \qquad (3)$$

The DFTs are taken from $x_t(m) = w(m)x(m+t)$, which is the windowed input signal at time-point t. The phase can be isolated from the magnitude spectrum. as the imaginary part of the logarithm of the DFT, $imag\{ln(DFT)\}$. The derivative of the phase can be expressed in terms of the DFT of the signal:

$$\frac{\partial}{\partial t}\theta(\omega,t) = \frac{\partial}{\partial t} imag\left\{\ln[DFT(x_t(m),k)]\right\} = \\ = imag\left\{\frac{1}{DFT(x_t(m),k)} \times \frac{\partial}{\partial t} DFT(x_t(m),k)\right\} \quad (4)$$

All that is necessary is to find the time derivative of the DFT, which can be defined as:

$$\frac{\partial}{\partial t} DFT(x_t(m),k) = \\ = j\omega DFT(x_t(m),k) + DFT(x'_t(m),k) \quad (5)$$
$$(with \quad x'_t(m) = -\frac{\partial}{\partial m} w(m)x(m+t))$$

Substituting back in (4) and using the definition of the IFD given in (2), the final formulation is obtained:

$$IFD(x(n),k,t) = \omega + imag\left\{\frac{DFT(x'_t(m),k)}{DFT(x_t(m),k)}\right\} \quad (6)$$

The IFD is formulated as a ratio of two DFTs, one taken from a windowed signal and the other using the same signal windowed by the negative derivative of the same analysis window. The derivative of the analysis window is generated by computing the differences between its consecutive samples. Since we are only using a single time-point for the instantaneous frequency estimation, the rotation that appears in (3) is not needed, unlike in the phase vocoder case.

In the Abe definition for the IFD, the formula is slightly rearranged (in fact this will be the formulation used in the present implementation). Here it is possible to see that the instantaneous frequency deviation *d(x(n),k,t)* from the bin centre frequencies, also known as the reassignment vector, is inversely proportional to the signal power at that bin (eq. 10). This follows from the fact that the larger the frequency deviation is from the bin centre, the lower the bin magnitude will be:

$$d(x(n),k,t) = imag\left\{\frac{DFT(x'_t(m),k)}{DFT(x_t(m),k)}\right\} = \\ = \frac{imag\left[DFT^*(x_t(m),k)DFT(x'_t(m),k)\right]}{\left|DFT(x_t(m),k)\right|^2} \quad (7)$$

This indicates that low magnitudes can have an effect on the accuracy of the analysis. However, in practice, since we are also taking the amplitudes from the magnitude analysis, this fact will not have any consequences. The IFD shares some common as aspects to the hopsize-1 phase vocoder described in [7], a variation on the difference methods, but has been shown to produced more accurate result [8].

As mentioned above, the IFD is a case of frequency reassignment, which is known to provide perfect localisation for sinusoids, impulses and chirps in the continuous case. In practice, a very small amount of errors are inevitably introduced due to the sampling process, which will also be present in any other analysis method. The IFD has also been shown to be superior to a number of other frequency estimation methods for sinusoidal analysis, including the standard phase vocoder [9].

3. ANALYSIS METHODS

For spectral frame generation in the amplitude-frequency format, we will be using the modulus of the DFT of a windowed frame to obtain the magnitudes, as well as using the IFD to estimate the frequencies. In addition, the analysis will output the unwrapped phases, which may be used in resynthesis. The analysis process is outlined in figure 1.

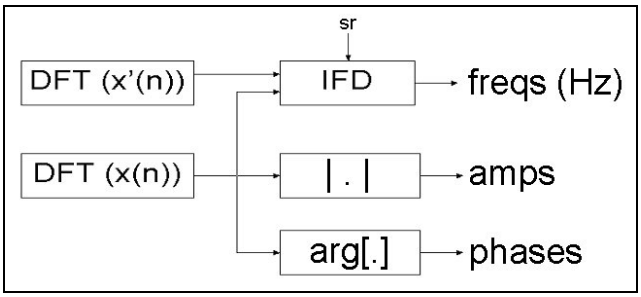

Figure 1. The IFD + magnitudes/phases analysis process.

3.1. Partial Tracking

In two of our proposed methods, the spectral analysis is followed by partial tracking, which will look for peaks in the spectrum and generate a number of tracks as output. This way we will be moving from a bin frame to a partial track format. Using the IFD, as well as magnitudes and phases from the analysis method described above, it is possible to employ a peak-picking algorithm to isolate partials. These can be organised into spectral tracks, containing frequency, amplitude and phase information. The phases are retained in the analysis, but will only be used in one of the resynthesis methods proposed here.

Partial track analysis relies on a simple principle. We start by identifying the spectral peaks at integral frequency points (bins), with a thresholding mechanism that eliminates low-amplitude components. Interpolated peak positions and amplitudes are then estimated using the magnitudes of the bins around the detected peaks. The exact values for the instantaneous frequencies are obtained by linearly interpolating bin values from the IFD input, whereas phase values are truncated. These will then, together with the amplitude, form a 'track', linked to each detected peak, to which a unique ID will be assigned.

The track will only exist as such if there is some consistency in consecutive frames, ie. if there is some matching between peaks found at each time-point. When peaks are short-lived, they will not make a track. Conversely, when a peak disappears, we will have to wait a few frames to declare the track as finished. These two analysis parameters are defined as the minimum points for a track and the maximum gap between points. Most of the process in partial analysis is one of track management, which accounts for the more involved aspects of the algorithm.

The partial analysis algorithm developed in this work has been implemented as a streaming process [10]. Here, track frames are

generated for every hop size of the analysed signal. Unlike other more straightforward implementations, we expect that only a single signal frame is present, or known, at a given time by the analysis algorithm, making it suitable, for instance, for realtime processing. Such implementation has a number of special details that are required for it to be successful . As an example, a record of past peaks is needed for the peak matching operation. In addition, a certain amount of delay is expected between the input and the output, which will depend on the analysis parameters, such as the transform size, and the minimum number of track points.

4. RESYNTHESIS

Three resynthesis methods have been developed for time-domain signal reconstitution. They all rely on the basic principle of additive synthesis using interpolating lookup table oscillators, defined by:

$$output(n) = \sum_{k=0}^{N} a(k,n)\cos(\theta(k,n)) \qquad (8)$$

where *k* is in the partial resynthesis case the track number or ID and in the bin frame case, the frequency bin index.

4.1. Linear amplitude and frequency partial track resynthesis

As discussed above, the partial tracking analysis will output tracks made up of frequencies and amplitudes. This is used for additive re-synthesis of the signal or of portions of its spectrum. A typical method involves the use of the frequency parameter to calculate the varying phase used to drive an oscillator and the amplitude to scale its output. The parameters can be interpolated linearly on a frame-by-frame basis, which was found to be efficient and sufficiently precise for many applications. The resynthesis parameters are calculated as indicated by eqs. 9 to 11.

$$\theta(k,n+1) = \theta(k,n) + 2\pi f(k,n)/sr \qquad (9)$$

$$f(k,n) = f(k,t) + [f(k,t+n) - f(k,t)](n-t)/h \qquad (10)$$

$$a(k,n) = a(k,t) + [a(k,t+n) - a(k,t)](n-t)/h \qquad (11)$$

where *a(trk,n)* and *f(trk,n)* are the interpolated amplitudes and frequencies, *sr* is the sampling rate, *k* is the track ID, *h* is the hopsize and *t* the analysis time-points.

The additive resynthesis will use the analysis track frames to drive a bank of cosine wave oscillators. Since this is a streaming process, we will use the track IDs to match tracks between analysis frames, in order to perform properly the interpolation of amplitude and frequency. It is also important to make sure that, when a track is created/destroyed, an amplitude onset/decay is created. This will help avoid problematic discontinuities in the resynthesised signal. In the implementation used for this work, the oscillator bank will employ interpolating table lookup oscillators.

4.2. Cubic interpolation partial track resynthesis

In situations where linear interpolation is not providing enough quality, we can apply a modified cubic interpolation algorithm. The original method for cubic phase reconstitution, as described in[11] does not have the flexibility to support important musical signal processes such as timescale and frequency transformations. We present here some important modifications of this method, which will allow these process to be implemented.

The original cubic interpolation of phases takes frequencies and phases at two time-points and interpolates the phase for oscillator resynthesis so that the initial and final phase values are matching. With the original implementation of this method, it was found that timescale and pitch transformations were only possible in certain restricted cases. We have proposed a modification that incorporates the addition of a small phase offset, which correctly modifies the target phase value, rendering the process possible. The interpolation procedure is defined by:

$$c_1 = 2\pi f(k,t) \qquad (12)$$

$$c_2 = \frac{3}{[h/sr]^2} \times \left[\Delta\theta(k,t) - \frac{2\pi h}{3sr}(2f(k,t) + f(k,t+n)) \right] \qquad (13)$$

$$c_3 = \frac{1}{3[h/sr]^2} \times [2\pi(f(k,t+n) - f(k,t)) - 2c_2(h/sr)] \qquad (14)$$

$$\theta(k,n) = \theta(k,t) + c_1\left(\frac{n-t}{sr}\right) + c_2\left(\frac{n-t}{sr}\right)^2 + c_3\left(\frac{n-t}{sr}\right)^3 \qquad (15)$$

where, as before, *f(t,k)* and *θ(t, k)* are the instantaneous frequencies and phases at the successive time-points *t* and *Δθ(t, k)* is the difference between the successive phase values at each time-point. The amplitude is interpolated linearly as in eq.11. This method allows for a more precise reconstitution of the phase than most other existing methods. In consequence, we have an enhanced quality resynthesis of transformed musical signals.

4.3. Cubic phase bin frame resynthesis

The third method proposed in this work is based on the cubic interpolation algorithm described above, with a small alteration. It takes bin frames of amplitude-frequency pairs in the same format as produced by, for instance, the phase vocoder, plus the unwrapped phase at each bin. In this work, this trio of parameters is generated by the IFD and magnitude-phase analysis, as described before.

This method has the practical advantage that it does not require the step of partial tracking and it can be plugged into a phase-vocoder-based transformation process. In fact, a phase vocoder analysis implementation (with the extra phase output) can be a substitute for the IFD and any data stored in that format can be resynthesised by this method. In practice, the IFD, as discussed above will offer a better frequency analysis, so it will be our preferred method.

5. TRANSFORMATION METHODS

5.1. Timescale modifications

Frequency-independent timescale transformations can be implemented with all the analysis-resynthesis methods described above. After IFD/magnitude and partial tracking (depending on the resynthesis method), the time-domain signal reconstitution can be made to proceed at any time rate, since the frequency-domain data contain basically the parametric values of amplitude and instantaneous frequencies. By spacing the resynthesis time-points to different hopsizes, time-scale modification (either stretching or compression) is produced.

If the interpolation to decimation ratio is kept at integral values, the cubic resynthesis methods will provide phase locking. This is particularly desirable in the bin-frame case, as lateral phase coherence is not an issue with partial track resynthesis. In this case, instead of scaling the resynthesis hopsize, keeping the original hopsize but changing the resynthesis frame rate is likely to be the preferred method of timescale modification. This is in fact the most common way of implementing time-scale modifications, used in many phase vocoder versions.

5.2. Frequency modifications

Time-independent modifications can also implemented with the proposed methods. Pitch shifting can be implemented with a simple frequency scaling control that will multiply the frequency of each oscillator in the resynthesis process. Frequency shifting, or partial stretching can also be performed by adding a constant frequency amount to each oscillator. In fact, any arbitrary frequency transformation function can be applied to any of the proposed methods, which is a process normally classed as frequency warping.

5.3. Other processes

In addition to the above, the analysis-synthesis methods described here has a variety of other potential applications, such as most of the ones described in [12]. In the partial tracking methods, for instance, basic noise-reduction can be performed by adjusting the analysis threshold parameter. Transients can be reduced by increasing the minimum number of analysis time points. Non-linear filtering can be achieved by limiting the number of analysis and/or resynthesis tracks. Other effects can be implemented by custom track processing classes. Furthermore, with the track analysis framework in place, it is possible to design and experiment new processes that take advantage of this data format.

With the bin-frame resynthesis method, it is possible to employ any standard frequency-domain processes already developed for phase vocoder signals. For example, transposition with formant preservation can be achieved by frequency scaling and bin reassignment, without any change to the magnitudes or by multiplying the reassigned magnitudes by the original magnitudes (also bin by bin, a case of spectral filtering).

6. IMPLEMENTATION DETAILS

In this work, the IFD, partial analysis and the three resynthesis methods have been implemented as classes in the SndObj library [13]. Musical signal transformation applications are created using a chain of SndObjs. The IFGram class implements the IFD + magnitude and phase analysis of the signal. Partial tracking is done by SinAnal objects, which takes an input from an IFGram object (or any signal in that format). Partial resynthesis is implemented by the AdSyn and ReSyn classes (linear and cubic interpolation respectively). In addition, the library also provides the original cubic phase interpolation, performed by SinSyn objects. The IFAdd class implements resynthesis from amplitude-frequency bin frames, as output by IFGram objects. The analysis objects can be connected as follows (`dec` is the hopsize, `ins` the input object):

```
IFGram  ifg(&win,&ins,1.f,fftsize,dec);
SinAnal part(&ifg,thresh,trks);
```

Track resynthesis then can be performed by linear interpolation, where `itp` is the hopsize, `scl` is amplitude scaling and `costb` is a cosine table object.

```
AdSyn synth(&part,trks,&costb,pitch,scl,itp);
```

Or, instead by cubic interpolation, where `str` is the stretch ratio, calculated from `itp:dec`.

```
ReSyn synth(&part,trks,&costb,pitch,scl,str,itp);
```

Finally, bin-frame resynthesis uses the IFGram object output, directly:

```
IFAdd  synth(&ifg,bins,&costb,pitch,scl,str,itp);
```

The advantage of implementing these methods in C++ as part of the SndObj library is evident in the fact that they can avail of all the facilities already present in that package. Several processes that manipulate amplitude-frequency data in a streaming fashion have already been implemented. Furthermore, the library provides a comprehensive, yet straightforward, framework for spectral signal processing.

Finally, some of the SndObj library code described above has also been ported to Csound5[14] in the form of a series of streaming spectral signal opcodes[15]. These form an initial set of partial track analysis-synthesis unit generators, design to add that facility to the csound language. It is expected that a number of track data manipulation opcodes will be added to the system, now that a framework for streaming partial analysis is in place.

6.1. Performance

Inevitably, additive resynthesis methods are in general less efficient than overlap-add ISTFT-based processes. However, in the case of partial track resynthesis, it is possible to reduce the computation time by adjusting the analysis threshold and limiting the number of tracks used. This way the overall efficiency can be determined by the user, possibly with a trade-off in terms of output quality or fidelity. Also, due to the partial analysis process, the computation requirement will vary depending on the complexity of the input signal. Sounds with more components will generate more tracks than simpler spectra. This will mean that the analysis process will have more tracks to manage, resulting in longer computation time. It is also possible to speed up computation by only analysing and/or resynthesising a maximum number of partials.

In terms of efficiency, the linear-interpolation partial track resynthesis method is the best. Tests have shown that it is possible to time-stretch a complex input soundfile in realtime, using current technology. Source sounds used in these tests varied from simple instrument sounds to recorded music. In addition, by resynthesising pre-analysed track data, stored in analysis files, it is possible to avoid the computation of the analysis step. These results have indicated the validity of implementing this technique as a streaming process, so that the development of realtime applications is facilitated.

The most computationally intensive resynthesis method is the IFAdd cubic interpolation method. Since it takes all the analysis bins for resynthesis, it relies on a large oscillator bank. It is possible however to limit the number of resynthesis bins using a variety of methods. These would range from bin limiting, which will reduce the output bandwidth, to fixed and adaptive thresholding operations. The great advantage of this method is that it is conceptually simple, making it easier to design efficiency measures for it.

7. RESULTS

In this work, we have mainly assessed the quality of the methods in relation to timescale modification, although some other tests have indicated the robustness of the techniques for other types of transformation. For time-stretching, in special, the quality of the output seems to be much superior to other more common techniques, particularly the phase vocoder. Partial tracking resynthesis, either with AdSyn or ReSyn, showed clean results for large time-stretch ratios. Based on the analysis of spectral peaks, they eliminate the typical artifacts associated with the phase vocoder technique. The IFAdd bin frame cubic resynthesis also provides a better result to the standard phase vocoder, due to the possibility of phase-locking.

A number of comparative tests were run using the methods proposed here against four different versions of the phase vocoder. Two of these were found in the Csound package: the original pvanal and Richard Dobson's pvocex, based on the Mark Dolson's CARL implementation. The other two were programmed in Pure Data, one of which used phase-locking. The methods described here offer considerable improvement in the quality of the output in relation to all of the standard phase vocoders, especially for large time-stretching factors. For these cases, only the phase-locked vocoder provided similar quality results. The best results were obtained using a transform size of 2048 points and, in the case of partial track analysis, a maximum of 500 tracks and a threshold of 0.003 (relative to the frame peak amplitude).

Particularly, IFAdd performs quite well in the resynthesis of attack transients after time-stretching. This can be explained by the importance of accurate phase reconstruction to transient response. However, in partial tracking resynthesis, due to the track analysis process, we will experience a clean slowing down of the attack transients, even with cubic interpolation. In any case, the audible quality of the result from both methods will be clearly superior to the phase vocoder. In comparison, the Csound implementations provided inaccurate stretching of short sounds, which transformed a single attack into two smeared strikes, completely distorting the note onset. Figs.2 to 5 compare the bin-frame cubic interpolation to the Pure Data standard phase vocoder and phase-locked vocoder (which uses Miller Puckette's version of the technique). While the IFAdd stretch preserves some of the original transients (although with some expected slowing down of the note onset), both phase vocoder examples smear the attack and almost completely lose its transients. The audible results show that the proposed methods clearly provides a superior result to other, more commonly available, techniques.

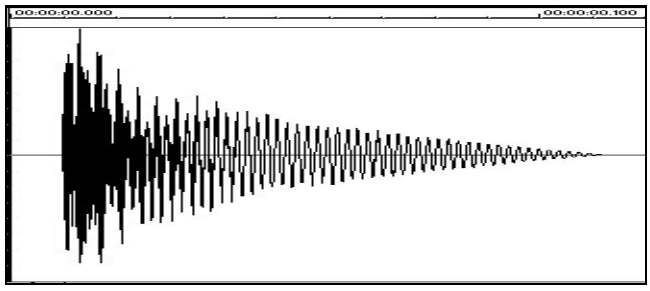

Figure 2. Short xylophone note (0.1 s)

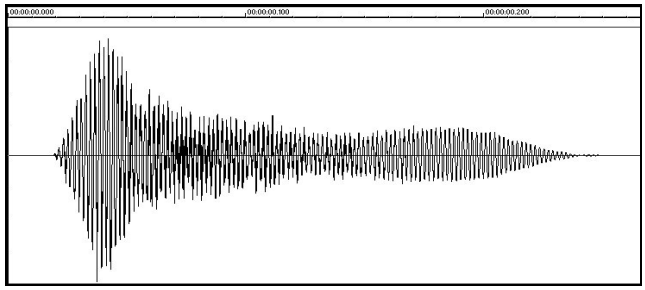

Figure 3. Standard phase vocoder time-stretch (2.5:1)

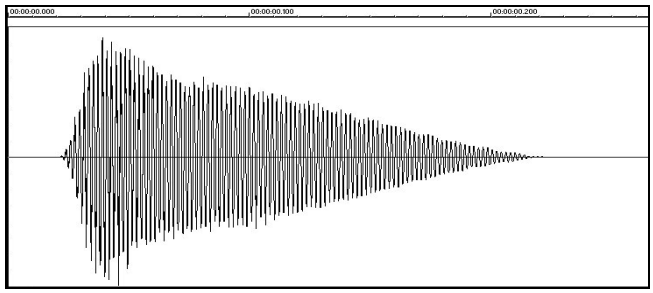

Figure 4. Phase-locked vocoder time-stretch (2.5:1)

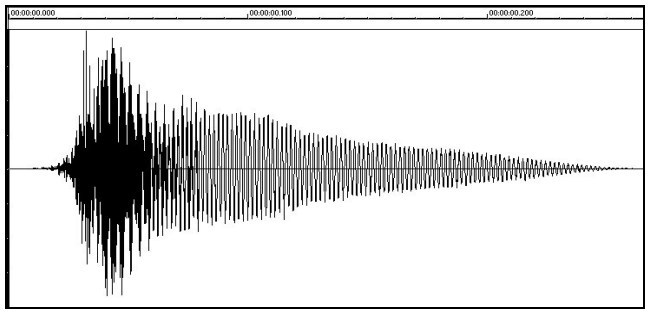

Figure 5. IFAdd time-stretch (2.5:1)

8. CONCLUSIONS AND FUTURE PROSPECTS

In this article, we have presented an alternative approach using the analysis-synthesis group of techniques for musical signal

processing. We have proposed a frequency analysis method based on the IFD and partial tracking that is robust and applicable to all musical sounds. In complement we have discussed three alternative methods of additive resynthesis based on linear and cubic interpolation. A number of sound transformation techniques can take advantage of these methods. The SndObj library implementation of the methods was discussed. A second implementation, in Csound5, has also been developed. We have presented very encouraging results for timescale modifications of musical signals with clean results and some degree of transient retention.

One of the aspects of these methods which will be further studied is the possibility of improving the cubic interpolation method to provide phase-locking at any interpolation to decimation ratios. Future work will also include the development of new transformation techniques, esp. for the manipulation of partial track data. This spectral format has some great potential for applications in sound transformation, both from a traditional signal processing perspective and using more unorthodox approaches.

9. REFERENCES

[1] T Wishart, *Audible Design*, Orpheus the Pantomine, York, 1995.

[2] M Dolson, "The phase vocoder tutorial", *Computer Music Journal, 10(4),* pp. 14-27, 1986.

[3] M Dolson, F Laroche, "Improved Phase Vocoder Timescale modification of audio", *IEEE Transactions on Speech and Audio Processing*, vol.7,issue 3, pp.323-332, May 1999.

[4] M Puckette, "Phase-locked vocoder". *Proc. IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, 1995.

[5] D Friedman, "Instantaneous-frequency distribution vs time: an interpretation of the phase structure of speech", *Proc. ICASSP*, pp 1121-4, 1985.

[6] T Abe et al, "The IF spectrogram: a new spectral representation," *Proc. ASVA 97*, pp. 423-430, 1997.

[7] M Puckette, J Brown, "Accuracy of frequency estimates using the phase vocoder", *IEE Transactions on Speech and Audio Processing 6 (2),* pp. 166-177.

[8] S Hainsworth, M Mclead, *Time-frequency reassignment: a review and analysis*, Technical Report, Cambridge Univ. Eng. Dept. CUED/FENG/TR.459, 2003.

[9] F Keiler, S Marchand, "Survey on extraction of sinusoids in stationary Sounds", *Proc. of DAFx02*, pp-217-221, 2002.

[10] J Timoney, V Lazzarini and T Lysaght, "New SndObj classes for sinusoidal modelling", *Proc. of DAFx02*, pp-217-221, 2002.

[11] R McCaulay, T Quatieri, "Speech Analysis/Synthesis Based on a Sinusoidal Representation", *IEEE Trans. On Acoustics, Speech, and Signal Processing, ASSP-34 (4)*, 1986.

[12] V Verfaille, P Depalle, "Adaptive Effects Based on STFT, Using a Source-Filter Model", *Proc. of DAFx04*, pp. 296-301 2004.1.

[13] V Lazzarini, "The sound object library", *Organised Sound 5 (1).* pp. 35-49, 2000.

[14] J ffitch, "On the Design of Csound5", Proceedings of the 3rd Linux Audio Conference, ZKM, Karlsruhe, Germany, 37-42

[15] V Lazzarini, "Extensions to the Csound Language: from User-Defined to Plugin Opcodes and Beyond", Proceedings of the 3rd Linux Audio Conference, ZKM, Karlsruhe, Germany, 13-20.